ESDSWG Reuse Discussions

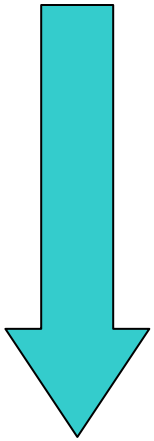
Hierarchy of Reuse -

Designing Systems to Promote Reuse

Reuse is Possible at Multiple Levels

- Function
- Class
- Function Library
- Class Library
- Subsystem
- System

Smaller,
Easier to Reuse



Larger,
Harder to Reuse

The challenge is to make it easier to reuse
at the subsystem and system levels

System and Software Developer Goals

- Lower Development Costs
- Speed Up Deployment
- Reduce Development Risks

For any software to be attractive for reuse,
it must help a developer meet these goals

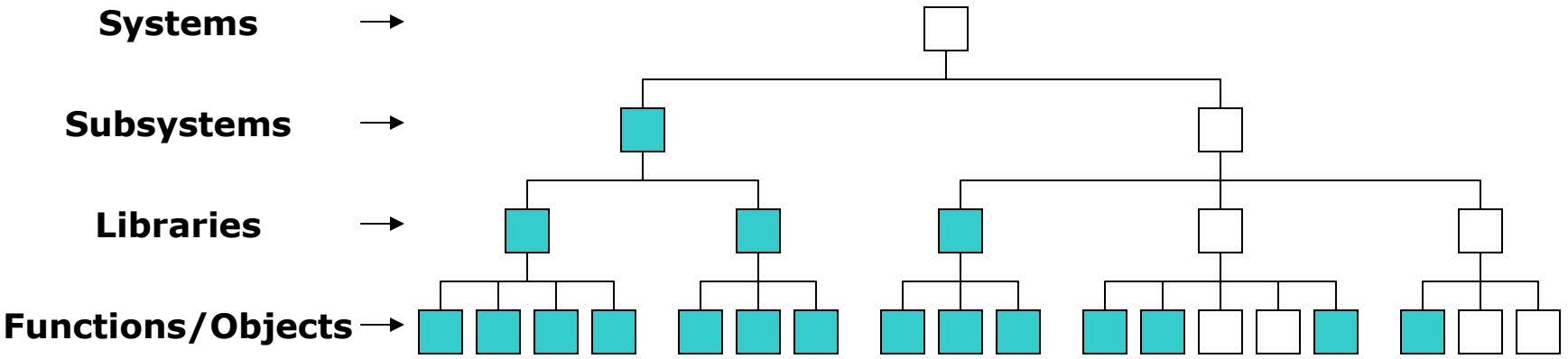
Attributes of Software that Make It Attractive for Reuse

- ☑ Source code available
- ☑ Well documented with clear APIs
- ☑ Fits the need without modification
- ☑ Carries no excess baggage
- ☑ Is Thoroughly Tested
- ☐ Is in Widespread Use
- ☐ Has Broad Community Support

☑ These attributes are under the control of developers

Hierarchical Reuse

- Use modular designs: each layer fills a well defined role and is fully documented and tested
  - Promotes reuse at multiple hierarchical levels



Reuse Don't Reuse

If every level is documented and tested, it has reuse potential

Maximizing Reuse Potential for a System

- Categorize and organize generic modules into reusable generic libraries
- Organize implementation specific modules in separate libraries
- Build in layers on only the generic modules to create generic subsystems when possible
- Introduce specific modules as far up the hierarchy as possible to create implementation specific subsystems and systems
- Document and test every module, subsystem, and system

Isolating the implementation specific modules from the generic modules of a system is the key to designing for reuse